Amendments to the drawings,

There are no amendments to the Drawings.

Remarks

Status of application

Claims 1-59 are pending in the present application. Claims 1-59 stand rejected on the basis of prior art. In view of the below remarks and amendments herein, reexamination and reconsideration of the claims are respectfully requested.

The invention

Applicant's invention provides for a strongly typed language binding for a remote method call and its parameter types, while deferring to runtime the actual processor binding. The methodology of the present invention separates the binding of a remote service into a specific language construct from the runtime code that accomplishes the remote method call. The same language binding may be used for any number of wire protocols and network transports. On the client side, a user/developer may implement an interface to the remote service by casting (instantiating) an instance of a generic interface class to create an interfaced object. This instantiated interfaced object includes an interface definition which is similar to a type library. This interface definition is an abstract description of a service that allows, for example, a client application to call a particular method or service running on a remote server without knowing the specific implementation details about how the method is implemented on the server. The compiler uses the instantiated interface object as if it was locally implemented to make the remote calls to the remote service. At runtime, this instantiated instance of the generic interface class (i.e., the interfaced object) dynamically proxies the call to the remote service when the remote service is called by a client application.

The dynamically generated proxy serializes the call on the remote service into a particular wire format specified in the interface definition of the service and sends call using a method of transport also specified in the interface definition. The specific wire format used may be any one of a set of possible implementations, as the present invention enables use of a number of different wire formats. The interface object also supports a number of modes of transport, such as TCP/IP, HTTP, or SMTP, for sending the packet to the remote service and receiving any response returned by the remote service. The

generic interface class of the present invention can be subclassed to provide support for specific wire protocols and wire formats. Using the present invention, a developer can create many different subclasses of the generic interface class to leverage its functionality for dynamic proxy generation, while allowing different transports and wire formats to be plugged in for use in a particular environment. When the remote method call from the client is received at the server, no service specific skeleton or stub is required. The serialization of the call made on the client side is reversed and the system converts the remote method call into a native call and dispatches that native call to the service.

Amendment to Specification

The Examiner objected to the Abstract as exceeding the limit of 150 words. Applicant has amended the Abstract so that it does not exceed 150 words, thereby overcoming this objection.

Claim objections

The Examiner objected to informalities in claim 39. Applicant has amended claim 39 to correct these informalities, thereby overcoming this objection. Applicant has also amended claims 2, 3, 10, 11, 48, 50, 52, and 56 to correct typographical errors.

Claim rejections under 35 USC Section 112

The Examiner has rejected claims 4 and 54 under 35 U.S.C. Section 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which Applicant regards as the invention. These claims have been amended to replace the phrase "may be" with the word "is" where required, thus overcoming the rejection under Section 112, second paragraph.

Prior art rejections

A. Claim rejections under 35 USC Section 102(e)

The Examiner has rejected claims 15-20, 22-24, 26-31, 33-40, 42-46, 48, and 49 under 35 U.S.C. 102(e) as being anticipated by US Patent 6,487,607 issued to Wollrath et

al (hereinafter "Wollrath"). Initially, it should be noted that the Examiner has not specifically listed claim 50 as being anticipated by Wollrath at paragraph 10 of the Office Action; however, the Examiner has indicated that claim 50 is rejected at paragraph 41 of the Office Action. Accordingly, Applicant assumes that the Examiner meant to reject Applicant's claim 50 under Section 102 as anticipated by Wollrath. If this assumption is incorrect, Applicant requests that the Examiner clarify the basis for rejection of claim 50.

The Examiner's rejection of Applicant's claim 15 as follows is representative of the Examiner's rejection of the above-listed claims as anticipated by Wollrath:

> As to claim 15: Wollrath teaches the invention as claimed including an improved method for making a service (e.g., (e.g., call or request 609; fig. 6) available to remote clients (e.g., client machine 601; fig. 6 & remote machines; col.4, lines 42-44), the method comprising:
>
> a. creating an interface definition (e.g., RMI 605; fig. 6) for a particular service (e.g., call or request for requesting invocation of remote object 608; fig. 6), the interface definition including runtime type information (e.g., the Java runtime system 516 includes the Java RMI system 518; col. 8, lines 44-46);
>
> b. implementing the interface definition as part of the particular service (e.g., RMI 605 receives a call or a request transmitted from RMI 602 for requesting invocation of a method of remote object 608; col.9, lines 57-61 & RMI 65 returns a response 610 using generic code 607; col. 10, lines 16-26);
>
> c. in response to a request (e.g., a call or request 609; col.9, lines 57-60) received from a remote client (e.g., a client machine 601; col.9, lines 54-55 and fig. 6) on the particular service, automatically converting the request into a native call on the particular service (e.g., Server machine 606 unmarshals the parameters for the operation given the types of the parameters specified in the method object; col. 10, lines 62-66); and
>
> d. converting return values resulting from the native call on the particular service into a format appropriate for return to the remote client and sending the return values to the remote client (e.g., marshals the return result(s) ... returns result(s) to the caller, client machine 601; col. 11, lines 7-14).

A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in the single prior art reference. (See, e.g., MPEP Section 2131.) As will be shown below, Wollrath fails to teach each and every element set forth in independent claims 15, 30, and 42 and the other claims listed above and therefore fails to establish anticipation of the claimed invention under Section 102.

At the outset, Applicant does not claim to have invented the notion of a client

which makes a remote procedure call on a server. At a high level both Wollrath's system and Applicant's invention include techniques for a client to make a remote procedure call on a service available from a remote server. However, Applicant's invention includes specific elements that distinguish it from the system of Wollrath as will be shown below. In particular, Applicant's invention provides for dynamically generating a proxy for making a remote method call on a remote service (e.g., Web service).

Applicant's approach does not require design time generation of code or type descriptions specific to the particular wire protocol, packet format, and remote programming interface being used. Instead, Applicant's approach provides for dynamically generating a proxy at runtime for calling the remote service. This is, for example, described in Applicant's specification as follows:

> At runtime, the application environment dynamically generates, at step 504, the proxy (i.e., the THHTPRio object) that makes the call on the Web Service. This is in contrast to prior art systems in which the proxy for remote method invocation typically must be generated when the application is compiled.
>
> (Applicant's specification, page 35, lines 21-24)

The dynamically generated proxy converts a remote method call by the client into the format specified in the remote service's interface definition and handles invocation of the remote service (Applicant's specification, page 35, lines 24-30). Applicant's independent claims 15, 30, 42 have been amended in an effort to more clearly articulate this distinction. Applicant's claim 15, as amended, includes the following claim limitations:

> creating an interface definition for a particular service, said interface definition including runtime type information;
> implementing said interface definition as part of said particular service;
> in response to a request received from a remote client on said particular service, using a dynamically generated proxy for sending said request to said particular service in a format specified in said interface definition and automatically converting said request into a native call on said particular service; and
> converting return values resulting from said native call on said particular service into a format appropriate for return to said remote client and sending said return values to said remote client.

(Applicant's claim 15, as amended, emphasis added)

Applicant's independent claims 30 and 42, as amended, also include similar limitations. Thus, Applicant's claimed invention provides for a dynamically generated proxy to send a request to a remote service in the appropriate format specified in the interface definition of the remote service. The proxy handles sending the call to the remote service, where it is converted into a native call on the particular service. Results of this native call on the service are then converted into an appropriate format and returned to the client. Wollrath does not teach a dynamically generated proxy which handles a client call on a remote service in the manner described in Applicant's claims. Wollrath's system simply provides for unmarshalling a remote method call received at the server as follows:

> Server machine 606 reads the object id and method hash (step 704), and it looks up the method object for the call using the method hash (step 705). Server machine 606 unmarshals the parameters for the operation given the types of the parameters specified in the method object (step 706). Step 706 may involve building a method table, which compiles values for particular methods and is initialized when a remote object is created and exported.
>
> (Wollrath, col 10, line 62 - col. 11, line 4)

Wollrath's approach provides for looking up a method object and unmarshalling the parameters from the stream of data (sequence of bytes) received at the server. The Examiner states that this teaches the corresponding elements of Applicant's claims as the fact that the call is unmarshalled at the server indicates that it must have been marshalled (converted) at the client side before it was transmitted to the server (Examiner office action, paragraph 58b, page 16). However, Applicant is not laying claim to the general notion of marshalling a remote method call at a client and unmarshalling the call at the server. Rather, Applicant's claimed invention provides specific features enabling a client application to more easily and efficiently invoke a remote service, without having to be concerned with specific implementation details of the remote service. More particularly, Applicant's claimed invention includes a dynamically generated proxy which

automatically converts a client call on the remote service into a format specified in an interface definition of the remote service and makes a call on the service. Applicant's careful review of Wollrath indicates that it does not teach these specific elements of Applicant's amended claims. In fact, the Examiner acknowledges that Wollrath is silent on a "dynamically generated" proxy as discussed below in more detail (Examiner office action, paragraph 49d, page 14).

Applicant's claimed approach also provides for creating an interface definition of a particular service available to remote clients which includes runtime type information. Applicant's dynamically generated proxy uses this interface definition for converting a remote method call into the appropriate format for invocation of the service. The Examiner relies on Wollrath at col. 8, lines 44-46 for the corresponding teaching of an interface definition of a remote service, the interface definition including runtime type information. However, the referenced portion reads as follows:

> The Java runtime system 516 includes the Java RMI system 518 and a JVM 520. Secondary storage device 504 includes a Java space 522.
>
> (Wollrath, col. 8, lines 44-46)

As shown, Wollrath simply refers to a Java runtime system including a Java RMI system and a Java Virtual Machine. It does not teach the specific limitations of Applicant's claims which describes an interface definition of a particular service which includes runtime type information.

Applicant's claimed invention provides for a dynamically generated proxy which performs the necessary conversion so that a remote method call from a client is converted into the required format for invocation of a remote service. The required format into which the remote method call is converted by the dynamically generated proxy is specified in the interface definition of the remote service. With Applicant's invention, a client application calling a remote service need not be concerned about implementation details of the remote service, as Applicant's dynamically generated proxy handles the details of converting and sending the call to the remote service in the appropriate format. Wollrath provides no teaching of a dynamically generated proxy which handles the

invocation of a remote service in the manner described in Applicant's amended claims. Therefore, as Wollrath does not teach or suggest all of the claim limitations of Applicant's independent claims 15, 30, and 42 (and other claims) it is respectfully submitted that the claims distinguish over this reference and overcome any rejection under Section 102.

B. Claim rejections under 35 USC Section 103

The Examiner has rejected claims 1-14, 21, 25, 32, 41, 47, and 51-59 under 35 U.S.C. 103(a) as being obvious over US Patent 6,487,607 issued to Wollrath et al ("Wollrath") in view of Pub. No. US 2002/0116454 of Dyla et al (hereinafter "Dyla"). The Examiner acknowledges that Wollrath does not teach dynamically generating an interface object for serializing data into a particular wire format for transmission or for transmitting Simple Object Access Protocol (SOAP) packets and adds Dyla for these teachings. The Examiner states that it would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine the teachings of Dyla and Wollrath because Dyla's teaching would have provided the capability for encoding the information in Web service request and response messages before sending them over the network (Examiner office action, paragraph 48, pages 12-13).

Under Section 103(a), a patent may not be obtained if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which the subject matter pertains. To establish a prima facie case of obviousness under this section, the Examiner must establish: (1) that there is some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings, (2) that there is a reasonable expectation of success, and (3) that the prior art reference (or references when combined) must teach or suggest all the claim limitations. (See e.g., MPEP 2142). As will be shown below, the references cited by the Examiner fail to meet the requisite condition of teaching or suggesting all of Applicant's claim limitations.

As discussed above, Applicant's claimed invention provides for a dynamically generated proxy to automatically perform the necessary steps to convert a remote method

call by a client on a remote service into the format required for invoking the remote service. Applicant's claimed invention includes specific limitations of obtaining an interface definition for the remote service and automatically converting the remote method call into a particular wire format and method of transport specified in the interface definition (see, e.g., Applicant's claim 1). A review of the Wollrath and Dyla references cited by the Examiner finds that they do not, either alone or in combination, teach or suggest these specific limitations of Applicant's claims.

Applicant's proxy is dynamically generated at runtime based on a generic interface class and an interface definition of a particular service available to remote clients (Applicant's specification, page 31, lines 26-29). The interface definition of the service includes runtime type information (Applicant's specification, page 31, lines 9-11). As discussed above, the Examiner relies on Wollrath at col. 8, lines 44-46 for the teaching of obtaining an interface definition of a remote service which includes runtime type information. However, Wollrath simply describes a Java runtime system and does not teach the specific limitations of Applicant's claims which describe obtaining an interface definition of a particular service which includes runtime type information. These limitations are, for example, included in Applicant's claim 1 as follows:

> An improved method for a client to make a remote method call on a service, the method comprising:
> obtaining an interface definition of a particular service available to remote clients, said interface definition including runtime type information;
>
> (Applicant's claim 1, emphasis added)

Applicant's dynamically generated proxy uses this interface definition of the particular service for converting a remote method call into the appropriate format for transmission to the service. This conversion includes specific substeps of converting the call into a wire format specified in the interface definition and using a method of transport also specified in the interface definition (Applicant's specification, page 35, lines 24-29).

The Examiner references Wollrath at col. 10, lines 16-22 for the corresponding

teachings of converting a call into a wire format and method of transport specified in the interface definition. As discussed above, Wollrath simply refers to unmarshalling of parameters of the method call at the server. However, the fact that a remote method call is marshalled at a client and unmarshalled at the server does not teach the specific limitations of Applicant's claims. Applicant's claimed invention includes a dynamically generated proxy that automatically converts a remote method call into a particular wire format specified in an interface definition of the service and sends the call using a method of transport also specified in the interface definition. These limitations are specified, for example, in Applicant's claim 1 which includes the following limitations:

> casting a generic interface class to said particular service for dynamically generating a proxy at runtime for making a remote method call on said particular service; and
> when a call is made by a client on said particular service, said proxy automatically making a remote method call on said particular service by performing the substeps of:
> converting said call made by said client into a converted call in a wire format specified in said interface definition; and
> sending said converted call to said particular service using a method of transport specified in said interface definition.
>
> (Applicant's claim 1, emphasis added).

Applicant's review of Wollrath finds no reference to converting a remote method call into a particular wire format or to using a particular method of transport specified in an interface definition of the remote service. Dyla does not cure the above-described deficiencies of Wollrath. The Examiner acknowledges that Wollrath does not teach dynamically generating a proxy at runtime and references Dyla for these teachings. The portion of Dyla referenced by the Examiner provides as follows:

> By default, WOLF adapters provide a Java Application Programming Interface (API) which makes use of a Remote Procedure Call (RPC) paradigm which uses XML as a communications technology. This is accomplished through the use of stubs and skeletons (terms well-known to the industry), which may be implemented through dynamically generated proxies on the client side.
>
> (Dyla, paragraph 35)

Although Dyla does mention dynamically generated proxies, it does not teach the specific limitations of Applicant's claims. Applicant's claimed approach includes dynamically generating a proxy at runtime for making a remote method call on a particular service in which the dynamically generated proxy converts the remote method call into a specific wire format and method of transport specified in the interface definition of the particular service. Applicant's careful review of Dyla finds no comparable teaching of a dynamically generated proxy which automatically performing the required conversion necessary to proxy a remote method call to a service using a wire format and method of transport specified in the interface definition of the service in the manner set forth in Applicant's claims.

Further distinctions are also provided in Applicant's dependent claims. For example, Applicant's claim 2 includes as claim limitations that Applicant's generic interface class "is subclassed to provide support for particular wire formats" (Applicant's claim 2). Applicant's claim 52 includes similar claim limitations. The Examiner references Wollrath, at col. 9, lines 61-67 for the teaching of a generic interface class which is subclassed to provide support for particular wire formats. However, the referenced portion of Wollrath reads as follows:

> Generic proxy 604 provides an advantage of not being type-specific so that it may invoke methods of varying types of remote objects. Table 1 contains a class definition written in the Java programming language for a generic proxy class by Netscape Communications Corp. and extended to allow the generic proxy to support invocation to methods among a set of interfaces.
>
> (Wollrath, col. 9, lines 61-67)

As shown, the referenced portion of Wollrath does not include the specific teaching set forth in Applicant's claim limitations of a generic interface class that is subclassed to provide support for particular wire formats.

All told, the Wollrath and Dyla references, even when combined, do not teach the specific limitations of Applicant's claims in which an interface definition is obtained for dynamically generating a proxy for handling a remote method call to a remote service. In

particular, neither Wollrath nor Dyla teach a dynamically generated proxy which converts a remote method call on a particular service into a wire format and method of transport specified in the interface definition of the particular service in the manner set forth in Applicant's claims 1 and 51 (and other claims). Accordingly, as the prior art reference(s), even when combined, fail to teach or suggest all the claim limitations, it is respectfully submitted that Applicant's claimed invention as set forth by these claims is distinguishable over the two references, and that the rejection under Section 103 is overcome.

Conclusion

In view of the foregoing remarks and the amendment to the claims, it is believed that all claims are now in condition for allowance. Hence, it is respectfully requested that the application be passed to issue at an early date.

If for any reason the Examiner feels that a telephone conference would in any way expedite prosecution of the subject application, the Examiner is invited to telephone the undersigned at 408 884 1507.

Respectfully submitted,

Date: June 23, 2005

Digitally signed by John A. Smart Date: 2005.06.23 13:59:22 -07'00'

John A. Smart; Reg. No. 34,929
Attorney of Record

408 884 1507
815 572 8299 FAX